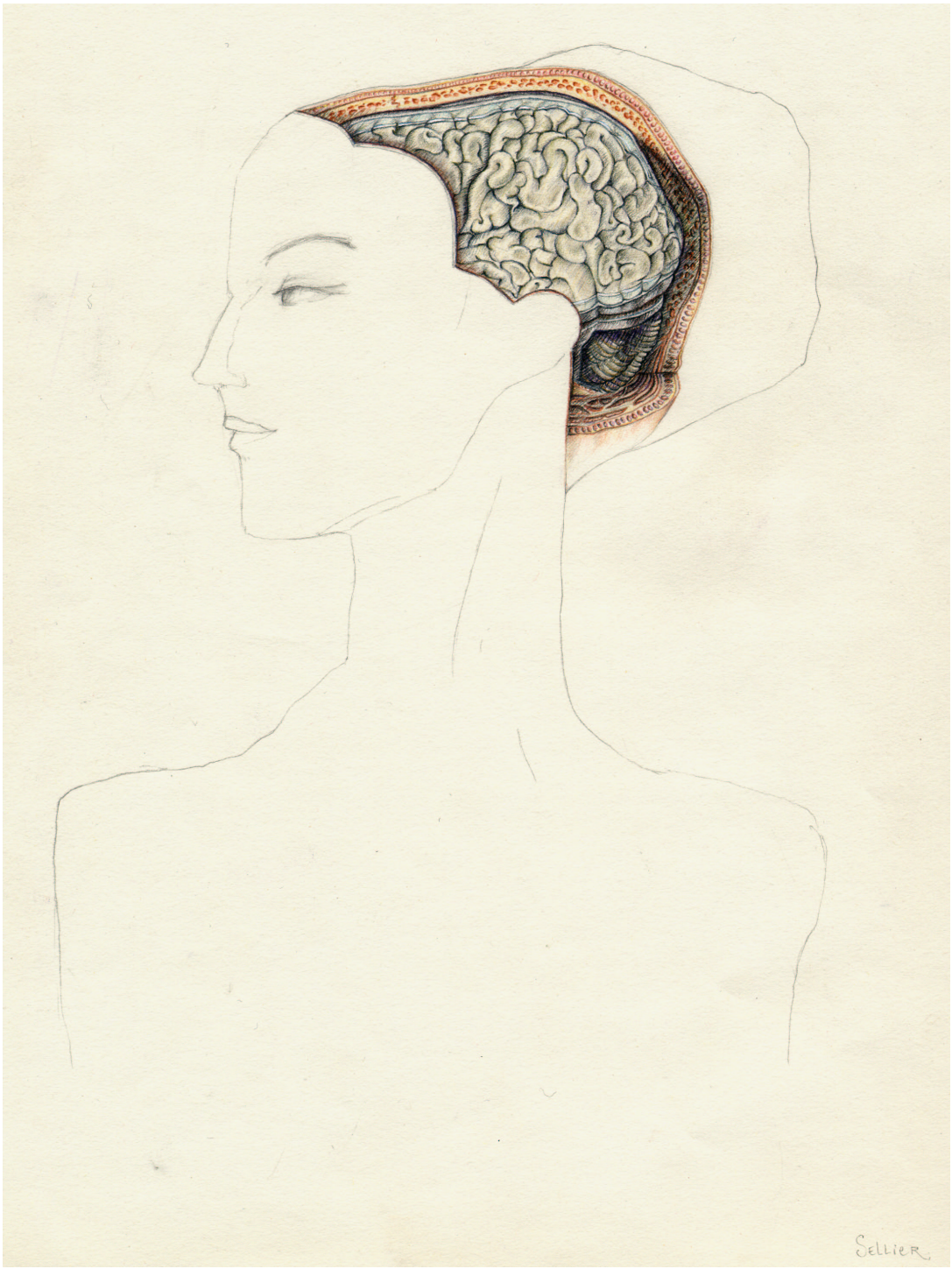
II

Extraction de Surfaces


Sellier

Chapter 4

Modèles Déformables pour l'Extraction de Surfaces en Imagerie Médicale

Résumé — Dans ce chapitre, nous revenons sur différentes méthodes d'extraction de surface en imagerie médicale. Partant du déjà connu modèle des *snakes* de *Kass, Witkin, et Terzopoulos* [82], en section 4.1, nous présentons des méthodes basées sur une représentation explicite de la surface, comme dans les travaux de *Delingette* [38]. Nous étudions ensuite en section 4.2 une représentation implicite de la surface, à partir des travaux de *Caselles, Kimmel, et Sapiro* [22]. Nous détaillons l'implémentation de ces contours actifs géodésiques dans la section 4.3, à partir de la méthode des Ensembles de Niveaux, développée et amplement détaillée dans le livre de *Sethian* [163]. On s'intéresse en section 4.4 à définir un modèle abstrait pour nos applications, et on poursuit par son implémentation à l'aide de modèle déformables explicites puis implicites. Nous présentons par la suite l'utilisation du *Fast-Marching* comme algorithme de segmentation dans la section 4.5, et nous détaillons les applications existantes de cette méthodologie à des problèmes classiques de segmentation en imagerie médicale 3D à la section 4.6.

Abstract — In this chapter, we recall the different methods used for surface extraction in medical imaging. Starting from the already studied *snakes* framework of *Kass, Witkin, and Terzopoulos* [82] in section 4.1, we mention methods based on explicit representations of the shape, as done by *Delingette* [38]. Thus we extend in section 4.2 to implicit representations of the shape, proposed by *Caselles, Kimmel, and Sapiro* [22]. We detail implementation of those Eulerian active contours in section 4.3, using the level-sets methodology developed extensively by *Sethian* [163]. We develop an abstract deformable model for our applications in section 4.4, and we follow by implementations of this framework with explicit and implicit deformable models. We further detail the use of the *Fast-Marching* method to segmentation tasks in section 4.5. And we detail several applications of this methodology to medical image segmentation problems in section 4.6.

4.1 Classical Active Contours

4.1.1 Definition

We recall the *Snake* model as introduced in [82], and already mentioned in chapter 1. For a general overview on deformable models, see [115]. See also a description and references in [30].

The classical energy of the model which will be minimized on $\mathcal{A}$ the space of all admissible curves, and has the following form:

$$\begin{aligned} E: \quad & \mathcal{A} \to \mathbb{R} \\ & \mathcal{C} \mapsto E(\mathcal{C}) \quad = \int_\Omega \frac{w_1}{2}\|\mathcal{C}'(v)\|^2 + \frac{w_2}{2}\|\mathcal{C}''(v)\|^2 + P(\mathcal{C}(v))dv \end{aligned}$$

where $\Omega = [0,1]$ is the parameterization interval. This model is used in the classical formulations of deformable models [32, 102].

In this formulation, each term appears as a potential acting on the shape. Thus the mechanical properties of the deformable model are controlled by two kinds of constraints:

- The internal potential: C' and C'' are the smoothing terms on the curve. They enable to control its regularity by means of w_1 which quantifies its rigidity and w_2 its elasticity;
- The external potential term P represents the likelihood. This "image" potential traps the curve towards the regions with desired attributes.

Figure 4.1[1] displays iterations of the heart segmentation in a 3D ultrasound image of the heart, with simplex meshes.

4.1.2 Drawbacks

The main drawbacks of the classical deformable model approach are:

- Minimization: The functional is non-convex, and one difficulty is to find a good local minimum. Spurious edges generated by noise may stop the evolution of the surface, giving an insignificant local minimum of the energy;
- Initialization: The user must specify an initial shape that is close to the goal, like a very precise polygon approximation, which may be tedious to draw;
- Topology changes: this method is unable to segment several objects simultaneously, and to merge different shapes;

One of the main issue in using deformable models is their initialization and minimization. The solution introduced in [163] allows to solve the global minimization of a problem. His approach considers a slightly modified problem: a curve is considered as an interface between two media, following a particular evolution equation. We will see that under precise assumptions, this front evolution efficiently builds a path between two fixed points, as detailed in [34].

[1] Slice of a 3D ultrasound dataset acquired with a multi-plane trans-oesophagus scan-head on a HDI 5000 ATL echograph.

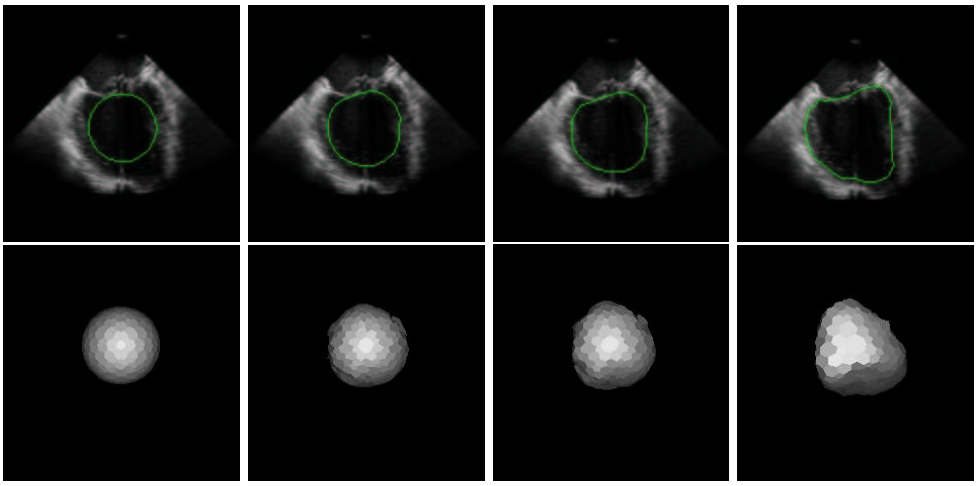


Figure 4.1. Samples of the segmentation of a heart in a 3D ultrasound image with simplex meshes [37] : Starting with a sphere interpolated with simplex mesh faces, we segment a heart in a 3D ultrasound image: first row shows the intersection of the mesh with a slice of the dataset, and second row is the 3D model at the same iterations.

4.2 Geodesic Active Contours

A geometric approach for deformable models was introduced by *Caselles, Catté and Dibos* [21] and *Malladi, Sethian and Vemuri* [113]. The basic idea of the geometric model is that the curve follows an evolution by expansion in the normal direction, with lower speed when the image force $P(\mathcal{C})$ is small. Hence the evolution of a planar curve C is in the direction of its normal only is given by

$$\frac{\partial \mathcal{C}(s,\tau)}{\partial \tau} = P(\mathcal{C})(\frac{\partial^2 \mathcal{C}}{\partial s^2} + w\ \mathbf{n}) = P(\mathcal{C})(\kappa + w)\mathbf{n}, \tag{4.1}$$

where s is the arc-length parameter of the curve $\mathcal{C}$, κ is the curvature, $\mathbf{n}$ is the unit normal. The constant term w is similar to the balloon force introduced in the *snakes* model [29] (and also related to the dilatation transform in mathematical morphology and the grass-fire transform [102]).

It was shown that the geometric snakes model handles topology changes better than the classical snakes when implemented with the level set approach for curve evolution proposed by *Osher and Sethian* [135, 158].

But, due to the formulation of the image force, it never comes to a complete stop, and heuristic stopping procedures are used to switch off the evolution process when an edge is reached. In equation (4.1), the geometric snake evolution is slower when the P is small but the curve does not necessary stop completely at the boundary, since it never reaches an equilibrium.

Given an initial curve $\mathcal{C}(s,0)$, the *geodesic active contours* is based on the planar

evolution equation

$$\frac{\partial \mathcal{C}(s,\tau)}{\partial \tau} = (P(\mathcal{C})(\kappa + w) - \langle \nabla P, \mathbf{n} \rangle)\, \mathbf{n}, \tag{4.2}$$

where s is the arclength. The ∇P term added, in comparison to the geometric model, is a projection of the attraction force $-\nabla P$ on the normal to the curve. This force balances the other term close to the boundary and causes the curve to stop there.

This introduction of ∇P, based on geometrical as well as energy minimization reasoning, leads to the "geodesic active contour" proposed by *Caselles, Kimmel and Sapiro* [23]. The geodesic active contours enjoy the advantages of classical as well as geometric active contours, since it handles topology changes and reaches an equilibrium which is similar to the classical snakes.

The curve evolution equation is then reformulated and implemented using the *Osher-Sethian* numerical algorithm [135]. Similar geometric models were also introduced in [85, 183, 165, 149] and extended to color and texture in [153].

4.3 *Level-Sets* Implementation of the Geodesic Active Contours

Let $C_o(p)$ be a closed initial parameterized planar curve in an Euclidean plane, and $C(p,t)$ the family of curves generated by the movement of $C_0(p)$ in the direction of its outward Euclidean normal vector $\mathbf{n}$. The speed of this movement is supposed to be a scalar function of the curvature κ:

$$\begin{cases} \frac{\partial C}{\partial t}(p) &= F(\kappa)\mathbf{n} \\ C(p,0) &= C_0(p) \end{cases} \tag{4.3}$$

A Lagrangian approach might be considered to implement the curve evolution according to the above equations in motion equations of the discretized positions of $C(p)$.

But this formulation has several drawbacks. When tracking the motion of the interface C propagating along its normal direction with velocity F, most numerical techniques rely on markers, breaking it up into points connected by segments, and moving each point with speed F (see figure 4.2-left). The solution is supposed to gain in accuracy if the number of points is increased. Problems arise if different parts of the front cross each other, or if the shape tries to break into two pieces (see figure 4.2-middle and the book of *Sethian* [163] for details) or if two shapes try to merge into one (see figure 4.2-right).

Therefore, the main drawback of this approach is that the evolving model is not capable to deal with topological changes of the moving front, and external procedures must be added to detect and deal with merging and splittings. These methods were developed for active contour models, namely *Topological Snakes* (T-snakes) in [114], for triangulated meshes in [97], and for their dual simplex meshes in [121]. There is no suitable difference approximation scheme for the Lagrangian implementation, due to the time-dependent parameterization of the curve model, thus leading to stability problems.

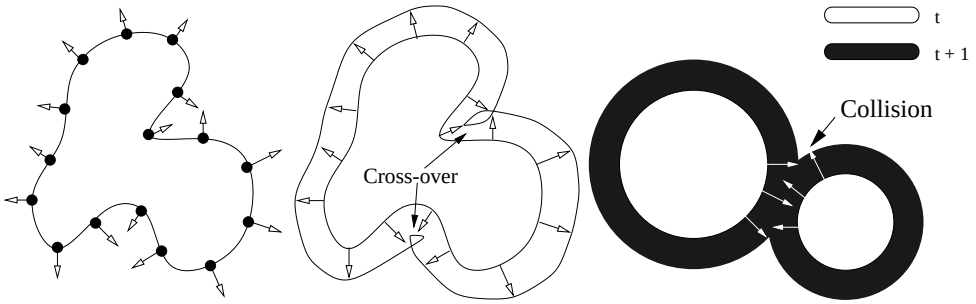


Figure 4.2. Markers methods and related problems: Left image illustrates the markers technique. Middle image shows the cross over which occur frequently. Right image shows problems of merging two contours.

These problems are handled very elegantly by the level set methodology, originally introduced by *Osher and Sethian* in [135], and now widely used in lots of applications, ranging from computer vision problems, to semiconductor manufacturing, shape from shading, and robotic navigation (see [163]). They embed the initial position of the moving interface $C_0(\mathbf{x})$ as the zero level set of a higher dimensional function ϕ (the signed distance to C_0, as shown in figure 4.3), and link the evolution of this new function ϕ to the evolution of the interface itself through a time-dependent initial value problem. At each time t, the contour $C(t)$ is given by the zero level-set of ϕ.

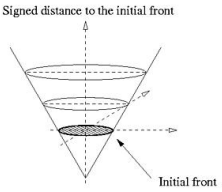


Figure 4.3. Embedding the contour in the signed distance: In this 2D example, the initial contour $C_0(x, y)$ is a circle, and $\phi(x, y, 0) = \pm d(C_0((x, y))$.

This condition states that

$$\phi(C(t), t) = 0 \Rightarrow \phi_t + \nabla\phi(C(t), t) \cdot \frac{\partial C}{\partial t} = 0 \tag{4.4}$$

Since $\frac{\partial C}{\partial t} = F\mathbf{n}$ and the outward normal vector is given by $\frac{\nabla\phi}{|\nabla\phi|}$, this yields the following evolution equation for ϕ given in [135]:

$$\begin{cases} \phi_t + F|\nabla\phi| = 0 \\ \phi(\mathbf{x}, 0) = C_0(\mathbf{x}) \end{cases} \tag{4.5}$$

4.3.1 Advantages of this formulation

There are several advantages associated with the *Level-Sets* paradigm:

1. This formulation remains unchanged in higher dimensions, as well for 2D curves, as for hypersurface in three dimensions (and higher). Therefore the embedded hypersurface will be denoted Γ in the following.

2. The evolving function ϕ remains a functions as long as F is smooth. As a consequence, topological changes in the evolving front $\Gamma(\mathbf{x}, t)$ are handled naturally, the position of the front at time t is given by the zero level-set $\phi(\mathbf{x}, t) = 0$ of the evolving function ϕ. $\Gamma(\mathbf{x}, t)$ can be several initial curves and it can break and merge as t advances. A 2D examples of fronts merging is shown in figure 4.4.

Figure 4.4. Easy handling of contour merging with *Level-Sets* : $\Gamma(x, y, t = 0)$ is the representation of the three curves. Initializing ϕ by the distance to these circles, and propagating ϕ with a constant positive speed in the outward normal direction, $\phi_t + \beta|\nabla\phi| = 0$, the three circles increase and merge. The different images represent the zero level-set of ϕ at several successive iterations.

3. Intrinsic geometry properties of the front are easily determined from the front itself, like the normal to the front $\frac{\nabla\phi}{|\nabla\phi|}$, and the curvature of the front $\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}$.

4. The evolution equation (4.5) can be approximated by efficient computational schemes with finite different approximation for the spatial and temporal derivatives. Explicit finite difference approach is possible, and other implicit and semi-implicit approaches have been already developed [65]

Explicit representation of the surfaces can also handle topology changes. Several attempt to achieve this tasks were developed independently.

- *McInerney and Terzopoulos* [114, 116] propose the *"T-snakes"* and *"T-surfaces"* with adaptive topology; the initial model is a triangulation that evolved according to a Lagrangian evolution equation, and the triangulation is resampled by computing its intersection with a tetrahedral griding of the image domain. By defining an "inside" and an "outside" region, the resampling handles topology changes when the surface self-intersect; this ad-hoc approach is limited to closed contours and surfaces.

- *Lachaud et al.* [98, 97] propose a very interesting technique based on the distance between each vertices of the triangulation. But knowing this distance between each pair of nodes is a huge computing task;

- *Montagnat and Delingette* [39, 40, 121] propose topological changes based on the simplex mesh surface representation [37]. This method that resamples the surface on a regular grid of the image domain works well in 2D and works with low computation times, but no 3D extension is available for the moment.

The different methods mentioned are all proposing methods to adapt the topology of their objects. This method has advantage of reducing the importance of the *a priori* on the final shape of the target of the segmentation process. Therefore, the topology of the model at initialization do not need to be the same than the model at convergence, thus reducing user interaction. But these methods are based on approximations, while the *Level-Sets* formalism handles naturally this problem. The *Level-Sets* model is evolved, and its zero level-set can be represented at several iterations, as shown in figure 4.4. But this implicit representation has one major drawback: it severely limits the possible interactivity of the user on the model, as mentioned in [120]. However, we will see in the next chapter that some basic interactions can be applied to this formalism.

4.3.2 Different Motions of the interface

The evolution of the segmentation is performed through the evolution of ϕ, which is done by a flow equation:

$$\frac{\partial \phi}{\partial t} + F|\nabla \phi| = \frac{\partial \phi}{\partial t} + \mathbf{V} \cdot \nabla \phi = 0 \tag{4.6}$$

if we consider the vector flow field $\mathbf{V}(\mathbf{x}, t)$ with $(\mathbf{x}, t) \in \Omega \times [0, +\infty[$, which evolves $\phi(\cdot, t)$.

Scalar flow

Any flow of the form:

$$\mathbf{V} = \beta(\mathbf{x}, t)\mathbf{n} \text{ with } \beta(\mathbf{x}, t) \in \mathbb{R} \tag{4.7}$$

will be referred as a scalar flow. Evolution under (4.6) with positive (resp. negative) values for β yields to a normal dilatation (resp. shrinkage) of $\phi(0, t)$. Figure 4.8 displays iterations of a front, initialized with the distance to a circle, evolving under an inflating term. In figure 4.5, ϕ is evolved with $\beta = -1$ in the expression of the scalar vector field of equation (4.7). Scalar flows lead to non-linear flow equations.

Vector Flow

Any flow of the form:

$$\mathbf{V} = \mathbf{U}(\mathbf{x}, t) \text{ with } \mathbf{U}(\mathbf{x}, t) \in \mathbb{R}^{\mathrm{d}} \tag{4.8}$$

and no dependency on ϕ will be referred to as a vector flow, or vector flow field. Evolution under (4.6) corresponds to passive advection of the level sets of $\phi(\cdot, t)$. Figure 4.6 displays iterations of a front, initialized with the distance to a circle, evolving under the influence of an advection flow only with $U(\mathbf{x}, y, t) = \mathbf{u} = (1, 1)$. The resulting contour is implicitly moved, in the direction of the vector $\mathbf{u}$. Vector flows lead to linear flow equations.

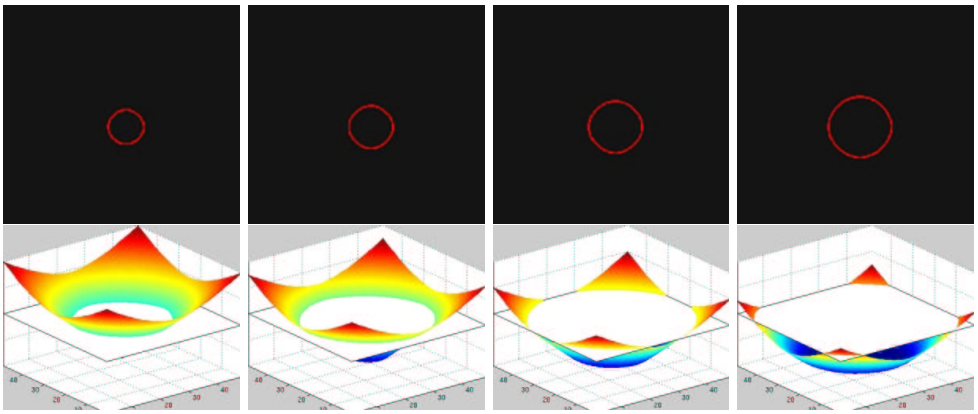


Figure 4.5. Inflating force: First row images are consecutive iterations of the *Level-Sets* evolution under external forces only; The external forces are built using the gradient of a left-ventricle image; Bottom row shows the zero level-set superimposed on the gradient magnitude at the same iterations.

Curvature motion

Any flow of the form:

$$\mathbf{V} = -\varepsilon(\mathbf{x}, t)\, \kappa_M(\mathbf{x}, t)\, \mathbf{n} \text{ with } \varepsilon(\mathbf{x}, t) \in \mathbb{R} \tag{4.9}$$

will be referred to as a curvature flow. Evolution under equation (4.6) with positive values for ε yields to a local regularization of $\phi(0, t)$. Negative values for ε lead to instabilities. Figure 4.7 displays iterations of a front, initialized with the signed distance to an initial curve, evolving under the influence of its curvature only, using the level set equation with a speed function of the form $F(\kappa) = -\kappa$

$$\phi_t = \kappa |\nabla \phi| = \left[\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right] |\nabla \phi| \tag{4.10}$$

If we let the iterations proceed, the zero level-set (in black) will tend to a circle, and shrink to a point before vanishing. Equation (4.10) applied to this image illustrates Grayson's theorem that all simple closed curves moving under its curvature must shrink to a point (as shown in [70]), regardless of its initial shape. This motion resembles a non-linear heat equation (see [163]) and smoothes large oscillations, and can be used in curve extraction as a diffusion term, to relax boundaries. Curvature flows lead to non-linear flow equations.

Composite flow

Any flow which is the sum of flows of the preceding types will be referred to as a composite flow. Figure 4.8 displays iterations of a front, initialized with the distance

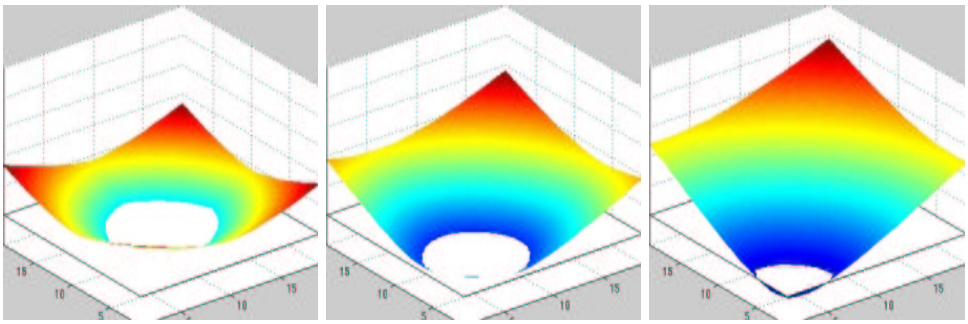


Figure 4.6. Advection flow: the level sets are represented using a colored ladder; the zero level-set is implicitly defined by the intersection between the levels and the white plan at height zero.

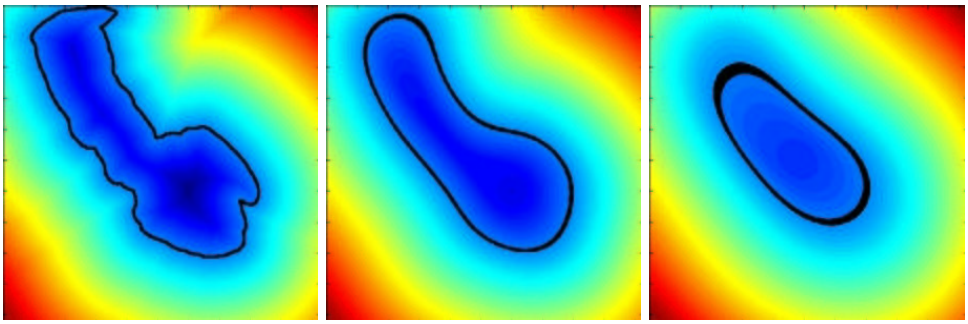


Figure 4.7. Curvature motion: the zero level-set (in black) and all the other levels evolve according to their curvature.

to a circle, evolving under the following equation:

$$\phi_t + g_I(1 - \varepsilon\kappa)|\nabla\phi| - \beta\nabla P \cdot \nabla\phi \tag{4.11}$$

where the equation contains the following terms:

1. an inflating force, as in equation (4.7), here defined by

$$g_I(\mathbf{x}) = \frac{1}{1 + |\nabla I_\sigma(\mathbf{x})|} \tag{4.12}$$

   where I_σ is the image convolved with a Gaussian kernel of size σ; this inflating force vanishes to zero in region of high gradients (i.e. near object boundaries)

2. a curvature term $-g_I\varepsilon\kappa$ which controls the smoothness of the iso-contours of $\phi(\cdot, t)$, originally introduced in [112];

3. an external force, which role is to attract the surface towards the boundary of the object of interest. This term is a vector flow which denotes a projection of an attractive force vector on the surface normal, in our case we use a potential field defined as in [22], by

$$P(\mathbf{x}) = -|\nabla I_\sigma(\mathbf{x})| \tag{4.13}$$

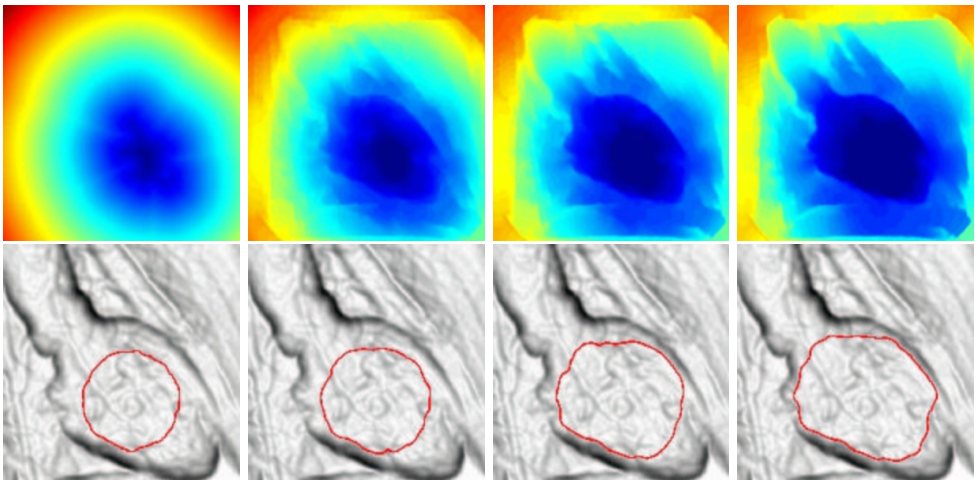


Figure 4.8. Composite flow: First row images are consecutive iterations of the *Level-Sets* evolution under the equation (4.11); the external forces are built using the gradient of a left-ventricle image; bottom row shows the zero level-set superimposed on the gradient magnitude at the same iterations.

4.4 Region-based forces

In this chapter we present a variational framework for the segmentation on the basis of the geodesic contour implementation of region-based forces. This framework has been used in the following parts of the thesis.

4.4.1 Partitioning the image domain

Region-based terms have already been included in active contour models [148, 27, 25, 24] We consider an *abstract deformable model*, which consists of a partition of the *image domain* Ω into three subsets: two open subsets $\Omega_{in}(t)$ and $\Omega_{out}(t)$ (the *inside* and the *outside* of the segmented object) and their common boundary $\Gamma(t)$. $\Gamma(t)$ is supposed to be as smooth as necessary (for example we may suppose that $\Gamma(t)$ is a locally Lipschitz-continuous hypersurface). This partition is a function of an evolution

parameter $t \in [0, +\infty[$ that will be called *time*[2]:

$$\begin{cases} \Omega = \Omega_{in}(t) \cup \Gamma(t) \cup \Omega_{out}(t) \\ \Gamma(t) = \partial\Omega_{in}(t) = \partial\Omega_{out}(t) \end{cases} \tag{4.14}$$

Figure 4.9 shows an example of 2D image, with the abstract deformable model as we would like it to look like after segmenting the image.

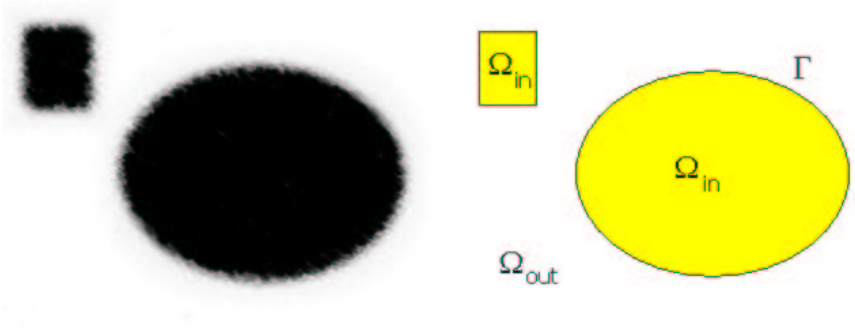


Figure 4.9. Abstract Deformable Model: left image is the 2D synthetic data; right image is the 2D abstract deformable model.

4.4.2 Region descriptors

Region descriptors are real functions of the image intensity I, and of Ω_{in} and Ω_{out}. We consider:

$$\begin{cases} k_{in}(\mathbf{x}, t) = -\log P_{in}(\mathbf{x}, t) \\ k_{out}(\mathbf{x}, t) = -\log P_{out}(\mathbf{x}, t) \end{cases} \tag{4.15}$$

where $P_{in}(\mathbf{x}, t)$ (resp. $P_{out}(\mathbf{x}, t)$) is the probability that x is in $\Omega_{in}(t)$ (resp. $\Omega_{out}(t)$).

Gaussian descriptors were first introduced by *Zhu and Yuille* [196]. They are "fully automatic" in the sense that no user-defined parameter is necessary to their definition. They are based on region probabilities defined by Gaussian distributions:

$$P_{in}(\mathbf{x}, t) = \frac{1}{\sqrt{2\,\pi}\,\sigma_{in}(t)}\ \exp(-\frac{(I(\mathbf{x}) - \mu_{in}(t))^2}{2\,\sigma_{in}^2(t)}) \tag{4.16}$$

A similar expression is used for the definition of $P_{out}(\mathbf{x}, t)$. Here $\mu_{in}(t)$ and $\sigma_{in}^2(t)$ are respectively the mean and the variance of the image intensity over $\Omega_{in}(t)$:

$$\mu_{in}(t) = \frac{\int_{\Omega_{in}(t)} I(\mathbf{x})\,d\mathbf{x}}{\int_{\Omega_{in}(t)}\ d\mathbf{x}}\ ,\ \sigma_{in}^2(t) = \frac{\int_{\Omega_{in}(t)} (I(\mathbf{x}) - \mu_{in}(t))^2\,d\mathbf{x}}{\int_{\Omega_{in}(t)}\ d\mathbf{x}}$$

This means that the histograms of the intensity in $\Omega_{in}(t)$ and $\Omega_{out}(t)$ are *modeled* by Gaussian distributions. Those descriptors were also used extensively when *Paragios*

[2]this "artificial" time has nothing to do with the notion of physical time.

introduced the concept of *Geodesic Active Regions* [137], for unsupervised image segmentation, supervised texture segmentation [139], and detection and tracking of moving objects [138].

4.4.3 Boundary descriptors

The boundary descriptor k_b, which is sometimes called *edge indicator*, is usually a real function of ∇I. We took the classical expression inherited from geodesic active contours (see for example *Caselles et al.* [23] or *Paragios* [137]):

$$k_b(\mathbf{x}) = \frac{1}{1 + \frac{\|\nabla I(\mathbf{x})\|^2}{\gamma^2}}$$

where γ is a user-defined parameter.

To make this descriptor "fully automatic", it is possible to use the mean gradient magnitude in the entire image and set:

$$\gamma = \frac{\int_{\Omega(t)} \|\nabla I(x)\| \, d\mathbf{x}}{\int_{\Omega(t)} d\mathbf{x}}.$$

This choice gave very good results on almost all the images we segmented.

Among other possibilities for boundary-based descriptors, we must mention the excellent work of *Xu and Prince* [189, 191, 190]. The construction of a new gradient information, using a somehow anisotropic diffusion technique to extend the local gradient information in the whole image, has been recently used together with the *Level-Sets* formalism in [140].

4.4.4 Segmentation as an optimization process

Composite energy functional

We see the process of segmentation of the image as the minimization of an *energy functional*, also called *objective function*. We decided to use a composite energy functional which comprises *region-based* and *boundary-based* terms. We took:

$$J(t) = \zeta \int_{\Omega_{in}(t)} k_{in}(\mathbf{x}, t) \, d\mathbf{x} + \zeta \int_{\Omega_{out}(t)} k_{out}(\mathbf{x}, t) \, d\mathbf{x} + \eta \int_{\Gamma(t)} k_b(\mathbf{x}) \, d\sigma \qquad (4.17)$$

where $d\mathbf{x}$, $d\sigma$, ζ and η are respectively the Lebesgue measures of $\mathbb{R}^{\mathrm{d}}$ and $\Gamma(t)$, and two user-defined positive scalar parameters.

The integrands in 4.17 are called *descriptors*, and contain all of the information that is being extracted from the image. The functions k_{in} and k_{out} are the *region descriptors*, and k_b is the *boundary descriptor*. The punctual values of the region descriptor k_{in} (resp. k_{out}) are supposed to be all the smaller as the probability to be in the interior (resp. exterior) of the object to be segmented is high. Similarly, the punctual values of the boundary descriptor are supposed to be all the smaller as the probability to be near a physical contour in the image is high.

The scalar parameters ζ and η may be adjusted to give more weight to the region-based integrals or the boundary-based integrals. The choice of ζ and η depends on the application specificities (quality and contrast of the images, image scale[3], etc.) and of the confidence of the user in the different descriptors.

Minimization of the energy functional

The evolution of the partition is totally defined by the evolution of the interface Γ between $\Omega_{in}(t)$ and $\Omega_{out}(t)$. The evolution (or motion) of Γ is defined by a speed field $\mathbf{V}(p,t)$, where p is the parameter corresponding to the chosen parameterization of Γ:

$$\frac{\partial \Gamma(p,t)}{\partial t} = \mathbf{V}(p,t). \tag{4.18}$$

We recall that the evolution of a hypersurface under an intrinsic (i.e. independent of the parameterization of the hypersurface) speed field depends only on the normal component of the speed (see Keriven [84, p. 40]).

The minimization of J is done by a gradient descent defined by the evolution equation (4.18). Several proofs have been proposed for the derivation of J. The most correct approach can be found in [79]. It shows that the evolution equation (4.18) implies that the temporal derivative of J is given by:

$$\begin{aligned}\frac{dJ}{dt} = \quad & \zeta \int_{\Omega_{in}(t)} \frac{\partial k_{in}}{\partial t}\, d\mathbf{x} + \zeta \int_{\Omega_{out}(t)} \frac{\partial k_{out}}{\partial t}\, d\mathbf{x} + \\ & \int_{\Gamma(t)} (\zeta\,(k_{in} - k_{out}) + \eta\,(k_b\,\kappa_M(p,t) + \nabla k_b))(\mathbf{V}.\,\mathbf{n})\, d\sigma\end{aligned}$$

where $\mathbf{n}$ and κ_M are respectively the normal (directed from $\Omega_{in}(t)$ to $\Omega_{in}(t)$) and the mean curvature of the hypersurface $\Gamma(t)$. In the case of time-independent region descriptors, the speed that minimizes the energy functional is consequently given by:

$$\mathbf{V} = \zeta\,(k_{out} - k_{in})\,\mathbf{n} - \eta\,(k_b\,\kappa_M(\mathbf{x},t)\,\mathbf{n} + \nabla k_b). \tag{4.19}$$

The evolution equation (4.18) will be embedded into the *Level-Sets* formulation described in section 4.3, and the flow (4.19) will be implemented using the different flows detailed in section 4.3.2. Note that $\mathbf{V}$ is intrinsic since it depends only on intrinsic features of $\Gamma(t)$. In the case of time-dependent region descriptors, other additive terms appear in (4.19) because of the two first terms in $\frac{dJ}{dt}$. But in the particular case of the Gaussian descriptors defined by (4.15) and (4.16), it is relatively easy to prove[4] that the two first terms in $\frac{dJ}{dt}$ are null, and that the speed that minimizes J is still given by (4.19).

The segmentation process consists in applying the evolution equation defined by (4.18) and (4.19) to a user-defined initial condition:

$$(\Omega_{in}(0), \Omega_{out}(0), \Gamma(0)) = (\Omega^0_{in}, \Omega^0_{out}, \Gamma^0)$$

The result of the segmentation process is given by the "limit" of the triplet $(\Omega_{in}, \Omega_{out}, \Gamma)$.

[3] Gaussian pre-smoothing of the image.

[4] All the vector analysis theorems needed for the proof are recalled in [79].

4.5 Segmentation with *Fast-Marching* algorithm

If we consider in equation (4.5) the particular case of a motion equation with a speed function $F > 0$, hence the interface always moves outward. One way to characterize the position of the interface is to compute the arrival time $T(\mathbf{x})$ of the interface as it crosses each point. Embedding this hypothesis in the level-set framework, a new equation is derived that determines the evolution of the surface $T(\mathbf{x})$ given by

$$\begin{aligned} T(C(\mathbf{x},t)) = t \quad &\Rightarrow \quad \nabla T \cdot C_t = 1 \\ &\Rightarrow \quad \nabla T \cdot \left(F \frac{\nabla T}{|\nabla T|} \right) = 1 \\ &\Rightarrow \quad F \cdot |\nabla T| = 1 \end{aligned} \tag{4.20}$$

that can be interpreted as: the gradient of arrival time is inversely proportional to the speed of the interface. This equation is the stationary case of the Hamilton-Jacobi formulation for propagating fronts where the time is disappeared, and if the speed is a function of the position of the front only, the equation reduces to what is known as the *Eikonal equation* equation, extensively studied in part I.

This particular equation (4.20) has also been used for surface extraction, since it has the same advantages as the *Level-Sets* formulation, detailed in section 4.3. In [111], *Malladi and Sethian* use the *Fast-Marching* algorithm in order to give a fast and rough initialization to a costly segmentation *Level-Sets* formulation. The *Level-Sets* model is based upon a composite flow, as shown in equation (4.11), for the segmentation of the cortex. The result of this segmentation was popularized by being advertised on the front cover of the *American Scientist Journal* [162]. They use as a speed function the following expression

$$F(\mathbf{x}) = e^{-\alpha |\nabla I_\sigma(\mathbf{x})|}, \; \alpha > 0 \tag{4.21}$$

where the speed has values very close to zero near high image gradients of the smoothed image I_σ. This expression is input in the *Eikonal equation* equation

$$|\nabla T(\mathbf{x})| = \frac{1}{F(\mathbf{x})} \tag{4.22}$$

The *Fast-Marching* in three dimension (see section 2.1) is then employed to march ahead. This helps to construct very quickly a good initial guess on the final surface. Then they input the final function $T(\mathbf{x})$ as an initial condition $\phi(\mathbf{x}, t = 0) = T(\mathbf{x})$, and iterate equation (4.11) a few time-steps with finite difference approximation schemes.

In practice the marching method is employed to march until a fixed time or until the size of the heap does not change very much (see section 1.3.2 for details on the heap used in the *Fast-Marching* algorithm) between two successive time increments.

4.6 Medical imaging applications of the *Level-Sets*

The *Level-Sets* models have already been used for a wide variety of applications, among them stereo problem [50], image classification [152], tracking of moving objects

[78], or modeling deformations of solid objects [185] among others. But the most important set of applications of the *Level-Sets* has been done in medical imaging, where their interesting properties in handling topology changes are very useful for segmenting the very complex anatomical shapes.

4.6.1 Cortex segmentation

Since initial contributions on 3D segmentation of medical images using the *Level-Sets* models by *Malladi and Sethian* [110, 111], the very complex shape of the cortical surface has attracted numerous contributions, due to the interesting properties of the *Level-Sets* formulation. One example of brain tissue segmentation is shown in figure 4.10, where the algorithm used is the *Fast-Marching* on the basis of the work presented in [111].

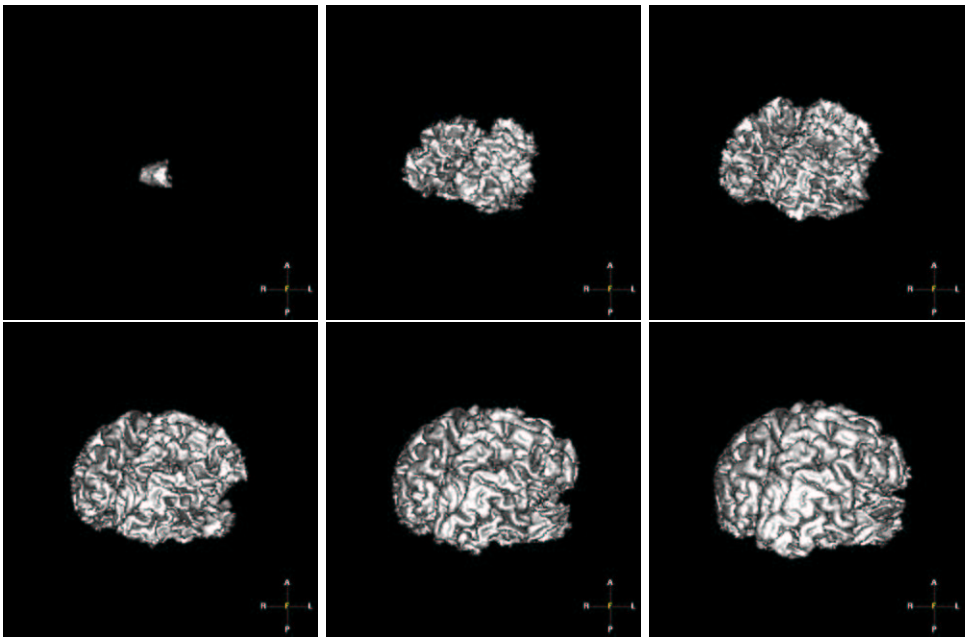


Figure 4.10. Segmentation result on the brain: On the basis of the model defined in [111], we segment the brain complex surface with the *Fast-Marching* algorithm, using a speed which is a simple distance function to an *a priori* mean value inside the white matter of the brain; formally $\mathcal{P}(\mathbf{x}) = \max\left(I(\mathbf{x}) - I_{mean}, 0\right) + w$.

In [67], authors proposed a fast implementation of the *Level-Sets* based on a semi-implicit formulation of the discretized evolution equation, based on similar use of these schemes for anisotropic diffusion filtering in [182].

An original method is presented in [7] where the authors propose to segment brain surfaces using sequentially a registration technique, and a *Level-Sets* model. However, it is not a real cooperation between both techniques, since the initialization

of any parameter of the *Level-Sets* are tuned on the basis of the first results obtained through the registration process. We will see in the following that the *Level-Sets* can cooperate with other fast segmentation techniques to enhance their preliminary results.

Several works have been developed on the basis of the simultaneous segmentation of the grey and white matter of the brain, using *coupled curve evolution equations* for segmenting both complex surfaces [193]. Main idea is to introduce a term relative to the distance between the two surfaces [194, 195]. In [68], this method is improved with a new scheme that computes the evolution of a new function which remains a distance function across iterations, and leads to better measurements of the distance between the two surfaces. And in [66] authors introduced the first geometric variational formulation for the coupled surfaces.

4.6.2 Brain Vessels

In the same region of the body, brain vessels extraction has attracted lots of attention from the computer vision community, with developments of very interesting dedicated *Level-Sets* implementations for thin tubular structure extraction.

We made a test on brain vessel extraction: the model is based on Gaussian region descriptors, and the initialization for k_{in} and k_{out} has been done using a **MDL** (minimum description length), with hypothesis that the image grey level information is a mixture of two Gaussian distributions. Descriptors are shown in figure 4.12. In figure 4.12 we show iterations of the segmentation of the brain vessels, in the

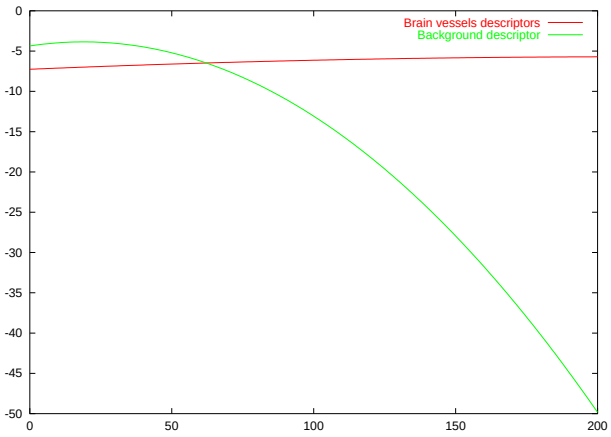


Figure 4.11. Gaussian region descriptors for brain vessels: assuming that the image is a mixture of two Gaussian distribution, the **MDL** classical computation of mean and variances has led to a model with ($\mu_{in} = 211.09, \sigma_{in} = 119.902$) and ($\mu_{out} = 18.9269, \sigma_{out} = 18.8822$).

dataset already displayed in figure 3.17, with the descriptors shown in figure 4.12. The propagation starts from one part of the vessels and propagates in the whole set.

Computing time for extracting the complete brain vessels is more than an hour, an a standard Sun workstation (300 MHz cpu).

In this field, *Lorigo et al.* [107] have developed flows using the Hessian information [108], co-dimension 2 evolution scheme for the extraction of thin curves in 3D [106], with application to the brain vessels. Same target was followed in [177] where the authors use a flow based on the divergence of the gradient in the image. Both works achieve extraction of thin curves where even *Level-Sets* classical formulation, as shown in figure 4.12, fails. However, the computing cost to achieve is too important. In the next chapter, we will settle another framework for segmentation in interactive computing time.

4.7 Summary

In this chapter we have explained an abstract deformable representation of the boundary between two regions, as used in [196]. The formulation of the evolution of the interface between the regions, defined with region descriptors, has been already studied for image segmentation in [137] with *Level-Sets* models. In the following we are going to develop improvements of this framework, introducing user interactivity on the interface, developments of algorithms to accelerate the speed of the computations, using the *Fast-Marching* first used for segmentation in [111]. On this basis we will show several applications of our framework.

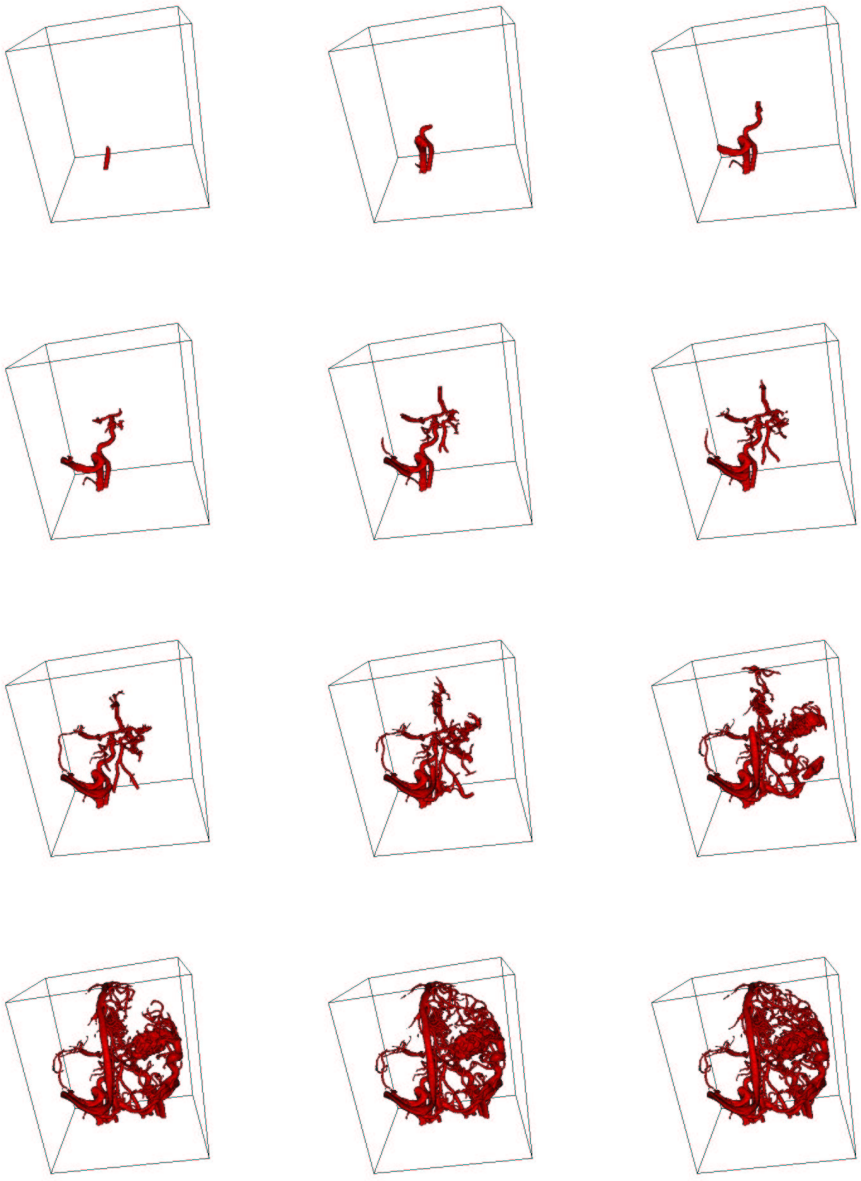


Figure 4.12. Extracting the brain vessels with region-based forces